02

Tratando los errores de API

Transcripción

[00:00] ¿Qué tal? Bienvenidos a la segunda clase de su curso. En el video anterior, en la clase anterior ya hemos visto cómo poner un poco mejor nuestra aplicación a nivel de código, siguiendo ya buenas prácticas, ya siguiendo más los estándares que se manejan normalmente en este mundo de Java y Spring.

[00:18] Por ejemplo, damos un vistazo al código, vamos a ver que ahora tenemos el ResponseEntity regresar los médicos. Ya no devolvemos directamente void, sino ahora usamos ResponseEntity, que es un wrapper para digamos encapsular la respuesta que le vamos a dar a nuestro servidor, por ejemplo aquí el ResponseEntity de la lista de los médicos.

[00:39] Aquí también de los datosRespuestaMedico, creamos nuestro DTO. Nuestra aplicación ya está conforme a algunas de las buenas prácticas que hay en el desarrollo de APIs usando Spring Boot, pero no es todo lo que tienen que aprender aún, porque si bien ya estamos siguiendo buenas prácticas, aun falta mucho, mucho, mucho por mejorar. Pero para eso estamos aquí.

[01:04] Ahora vamos a volver aquí a nuestro método. Y ya sabemos nuestro básico método para registrar, listar, actualizar, borrar y obtener. Pero la pregunta que yo les hago a ustedes aquí. ¿Qué pasaría si por ejemplo? Sabemos que el mundo no es perfecto, si yo tengo mi id 10, Spring me va a retornar el médico con el id 10.

[01:28] Pero como nada es perfecto en esta vida, ¿qué pasaría si yo le doy un id que no existe? Por ejemplo este de aquí. Para eso voy a encender mi servidor, porque aún no está corriendo, ejecuto mi servidor, perfecto, inicializa y ahora vamos a ver qué pasaría si yo ejecutó está llamada.

[01:53] Vemos que en efecto me está dando un error. ¿Pero qué error? Aquí dice 500 internal server error. Y de hecho 500 es uno de los códigos de error de HTTP. Como le comenté en el primer curso, en el rango de los 200 son los códigos de éxito, de 400 son errores de nivel de usuario o del cliente y de 500 para arriba son errores ya del lado del servidor.

[02:19] Ahora mi pregunta es ¿será bueno devolver toda esta información? ¿Y por qué se los pregunto? Miren aquí, tenemos información del timestamp. ¿Esto qué quiere decir? Hora y fecha en la que este error fue lanzado. ¿Esta información es buena? Sí, es buena, y me sirve. El estado: 500. El error fue un error interno del servidor ¿Por qué? Porque este id de médico no existe, entendible, pero la parte delicada comienza en el trace.

[02:49] ¿Qué es el stacktrace? Básicamente toda la excepción. ¿Por qué? Porque si seguimos aquí todo lo que nos dice él stacktrace al final lo que nos dice fue la excepción que lanzada a nivel de mi servidor. Incluso al final me dice: unable to find el médico, la entidad médico incluso me está revelando aquí el nombre de mi entidad que yo tengo con este id y el path que fue llamado.

[03:17] Entonces ahora yo tengo información sensible que está expuesta directamente a mi cliente, información a nivel de las clases que estoy usando, como mi clase médico como ya lo vi ahí y también aquí sobre el tipo de backend que yo estoy implementando. Yo puedo aquí saber que estoy usando JPA, puedo saber que estoy usando Spring y algunas otras cosas, me dan un vistazo de lo que está internamente en mi servidor.

[03:47] ¿Y esto en qué se traduce? En problemas de seguridad en información. Exacto. Son brechas de seguridad que quedan abiertas y cualquier hacker

puede explotar. Entonces aquí ya es nuestro primer contacto con lo que sería seguridad en sí, cómo podemos hacer para ocultar todo este stacktrace de la excepción a mi cliente de forma que no quede público.

[04:12] Ojo, yo no quiero decir que está mal devolver la información del error, lo que está mal devolver todo este stacktrace. Para mí, por ejemplo está perfecto devolver el timestamp, el status, el error y bueno el mensaje, pero el mensaje hasta podría ser un poco más personalizado también. ¿No les parece?

[04:30] Entonces vamos a personalizar esto un poco. Como les dije aquí el problema más grave es este stacktrace que está devolviendo aquí. Ahora yo necesito decirle a mi aplicación: "por favor, no devuelvas todo el stacktrace porque me estás exponiendo a futuras brechas de seguridad." ¿Cómo puedo hacer eso?

[04:52] Spring también nos da una facilidad para poder configurar esto. Si venimos al código, vamos a dar nuestro método para el get vamos a analizar un poquito qué es lo que hay. Por ejemplo, tenemos en nuestro ResponseEntity los datos, lo que él hace es llamar medicoRepository.getReferenceById.

[05:12] Y después en el dato médicos, lo que hacemos es copiar los datos en nuestra entidad para nosotros y retornar el DTO. Ya dijimos que lo que no queremos nosotros es devolver justo este estado de la excepción. ¿Qué hacemos para eso? Es una propiedad de Spring que tenemos que configurar.

[05:34] Y si les digo que es una propiedad de Spring que hay que configurar, ¿qué se les viene a la mente? Dónde es el mejor lugar para configurar propiedades a nuestro proyecto, el archivo application.properties.

[05:47] Entramos aquí y en esta oportunidad, por ejemplo, yo no recuerdo bien el nombre de la propiedad, porque tampoco es algo que se use todos los días. Entonces para eso, ¿qué hago? Voy a mi browser, está aquí, y voy a buscar aquí Spring Boot properties en Google.

[06:05] Digito Spring Boot properties, le doy enter y aquí vemos que la primera opción en este caso, a mí me sale la primera opción como Application Properties - Spring Boot, en docs.spring.io. Entro aquí y vemos aquí que tenemos todos los tipos de propiedades que Spring Boot maneja.

[06:28] Por ejemplo, tenemos propiedades sobre configurar JSON, por ejemplo, sobre spring data para distintos tipos de bases de datos, ustedes pueden darle un vistazo aquí a todos los tipos de properties que Spring está manejando, para manejo de transacciones, por ejemplo, para manejo de propiedades web y aquí está lo que nos interesa a nosotros: server properties. ¿Por qué?

[06:57] Porque a nivel de servidor lo que yo no quiero es mostrar mi stacktrace. Entonces voy a buscar aquí todo lo correspondiente a server.error, porque lo que yo quiero es tratar el error que mi server está retornando a mi cliente, y allí está, miren server.error.stacktrace. Entonces voy a copiar esta propiedad tal cual, tal cual está aquí.

[07:22] Regreso al proyecto, lo copio aquí y lo que aquí le voy a asignar siguiendo la documentación es never. ¿Por qué? Porque significa que yo no deseo nunca compartir el stacktrace con mi cliente. Voy a guardar aquí y voy a ver si mi aplicación va a cargar automáticamente. Sí, cargó automáticamente perfecto, por el web tools que tengo, y vamos a ver qué es lo que sucede ahora.

[07:52] Voy a buscar nuevamente por este ahí y que no existe y perfecto, miren, ya el stacktrace ya no aparece. Me sale el error de mensaje, unable to find la entidad médico. Quizás es algo que yo estoy dispuesto a aceptar este tipo de error, porque es más o menos entendible lo que me está diciendo, pero ya no sale el tacktrace el error. Entonces, primera parte de nuestra resolución de errores completa.

[08:21] ¿Qué es lo que está faltando aquí? Lo que está faltando aquí es tratar este código de aquí, porque como les dije, errores en el rango de 500 son errores a nivel de mi servidor, pero si yo busco por algún médico que no existe,

¿es error del cliente que pregunta por algo que no existe o es que mi servidor ha fallado por buscar un médico que no existe? Vamos a responder esa pregunta en el siguiente video.